Introduction

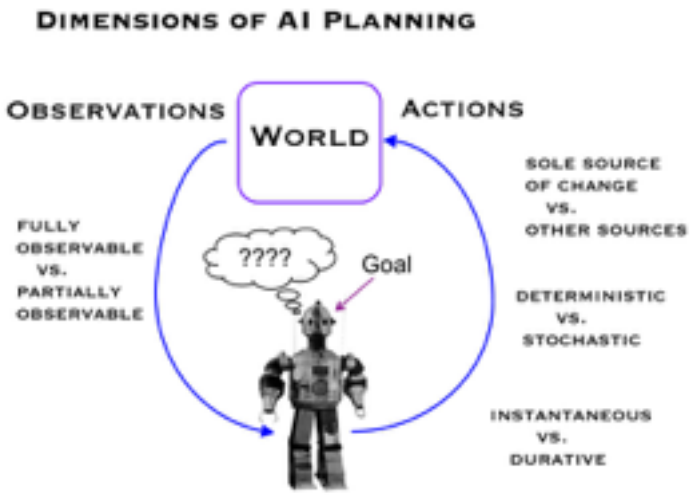
Artificial Intelligence (AI) has been studied for decades and is still one of the most elusive subjects in Computer Science. This partly due to how large and nebulous the subject is. AI ranges from machines truly capable of thinking to search algorithms used to play board games. It has applications in nearly every way we use computers in society.

Automated planning studies the problem of reasoning about actions to achieve goals or to maximise a reward—has been an important area of artificial intelligence (AI) research since the field began and this prominence is not difficult to explain.

First, the need to plan is pervasive; to a greater or lesser extent, all problems can be characterized as planning problems: how should one act (bring resources to bear) to change an existing state into a more desired state? The ability to act in a goal-directed fashion is critical to any notion of intelligent agency.

Second, planning is an extremely hard problem. Deterministic STRIPS planning (arguably the “easiest” type of propositional planning that is still capable of expressing interesting problems) is PSPACE-complete [13]; unrestricted probabilistic propositional planning in partially observable domains is undecidable [46].

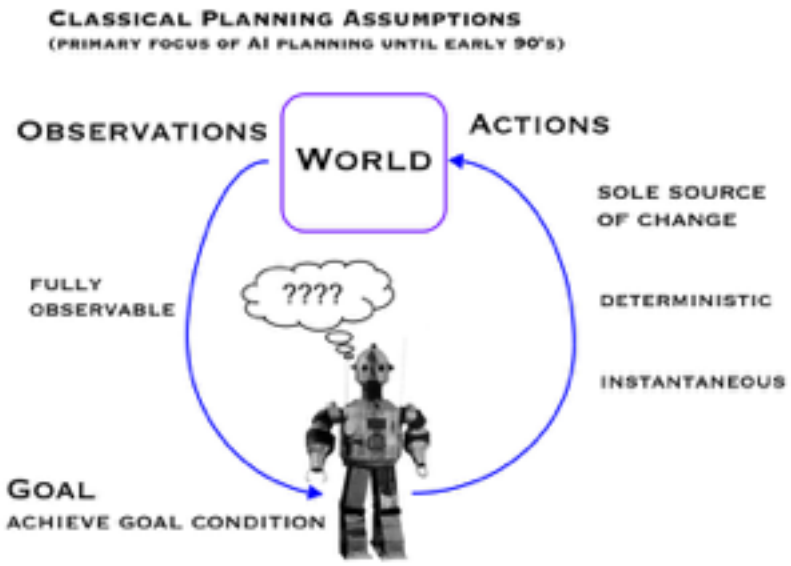
The picture below shows the various dimensions of AI Planning problems.



Classical Versus Non Classical Planning

Environments that are fully observable, deterministic, finite, static (change happens only when the agent acts), and discrete (in time, action, objects, and effects). These are called **classical planning** environments. In contrast, **nonclassical planning** is for partially observable or stochastic environments and involves a different set of algorithms and agent designs.

Up until the early 90’s the primary focus of AI planning was limited to classical fully observable domains with deterministic actions. And they admit no uncertainty : every aspect of the world that is relevant to the generation and execution of a plan is known to the agent in advance.



There has been a great deal of research in the past decade to create planning techniques that are capable of handling uncertainty in the environment (uncertain initial conditions, probabilistic effects of actions, and uncertain state estimation).

This fully observable and deterministic assumptions of classical planning greatly limits the practical applicability of AI planning techniques to many problem domains. But they are still very useful in the applicable domains for example logistics problems.

Planning is foremost an exercise in controlling combinatorial explosion. If there are p primitive propositions in a domain, then there are 2^p states. For complex domains, p can grow quite large. Consider that objects in the domain have properties (Location, Color, etc.) and relations (At, On, Between, etc.). With d objects in a domain with ternary relations, we get we get 2^(d3) states. We might conclude that, in the worst case, planning is hopeless.

Three Developments

The sections below outline three key developments in the history of AI that had a huge impact on the field of AI. We shall limit the discussion to developments in the domain of Search and Classical Planning. There are several other important developments that have happened in the last decade or so, and are still happening everyday in the field of Non-Classical Planning, but they would not be discussed in this report.

1. Informed Search and A* Algorithm

The move from un-informed graph search algorithms (such as breadth-first, depth first, uniform-cost) to informed search algorithms is the first important development in the field of AI. It tries to attack the first problem of handling combinatorial explosion encountered in un-informed search algorithms by incorporating knowledge of the problem domain in the form of **heuristics**.

In 1968, AI researcher Nils Nilsson was trying to improve the path planning done by Shakey the Robot[6], a prototype robot that could navigate through a room containing obstacles. This path-finding algorithm, which Nilsson called A1,

was a faster version of the then best known method, Dijkstra's algorithm, for finding shortest paths in graphs. A series of further refinements led to the A* algorithm which is now widely used in AI planning and search problems.

A* is an informed search algorithm, or a best-first search, meaning that it solves problems by searching among all possible paths to the solution (goal) for the one that incurs the smallest cost (least distance travelled, shortest time, etc.), and among these paths it first considers the ones that appear to lead most quickly to the solution. It is formulated in terms of weighted graphs: starting from a specific node of a graph, it constructs a tree of paths starting from that node, expanding paths one step at a time, until one of its paths ends at the predetermined goal node.

2. Planning : As a cross-fertilization of Search and Logic

The cross-fertilization of ideas from the two areas of Search and Logic has led to both improvements in performance amounting to several orders of magnitude in the last decade and an increased use of planners in industrial applications. We talked about **heuristics** in the previous section, but if the problem solving agent lacks autonomy and requires a human to supply a heuristic function for each new problem then its not a very good situation. On the other hand, if a planning agent has access to an explicit representation of the goal as a conjunction of subgoals, then it can use a *domain-independent* heuristic such as the number of unsatisfied conjuncts. In other words the agent should automatically be able to construct a sophisticated heuristic based on the available actions as well as the structure of the goal.

The preceding discussion suggests that the representation of planning problems—states, actions, and goals—should make it possible for planning algorithms to take advantage of the logical structure of the problem. The key is to find a language that is expressive enough to describe a wide variety of problems, but restrictive enough to allow efficient algorithms to operate over it. STRIPS (Fikes and Nilsson, 1971), was the first major planning system, and it illustrates the interaction of these influences. STRIPS was designed as the planning component of the software for the Shakey Robot project at SRI [6]. The action representation used by STRIPS has been far more influential than its algorithmic approach. Almost all planning systems since then have used one variant or another of the STRIPS language.

The proliferation of variants over time made comparisons needlessly difficult. With time came a better understanding of the limitations and tradeoffs among formalisms. The Problem Domain Description Language or PDDL (Ghallab et al., 1998) was introduced as a computer-parsable, standardized syntax for representing STRIPS, ADL, and other languages. PDDL has been used as the standard language for the planning competitions at the AIPS conference, beginning in 1998.

These methods are, however, inherently propositional and thus are limited in the domains they can express. (For example, logistics problems with a few dozen objects and locations can require gigabytes of storage for the corresponding CNF expressions.) It seems likely that first-order representations and algorithms will be required if further progress is to occur. Propositional logic is too puny to represent knowledge of complex environments in a concise way. First-Order-Logic on the other hand is sufficiently expressive to represent a good deal of common sense knowledge.

3. Problem Decomposition, Interleaving and the Concept of Partial Order Planning

A problem solver would be inefficient if it cannot take advantage of problem decomposition. Planners in the early 1970s generally worked with totally ordered action sequences. Problem decomposition was achieved by computing a subplan for each subgoal and then That is, the planner can work on subgoals independently, but might need to do some

additional work to combine the resulting subplans. Stringing the subplans together in some order. This approach, called **linear planning** by Sacerdoti (1975), was soon discovered to be incomplete.

Sometimes it is possible to solve a problem efficiently by recognizing that negative interactions can be ruled out. We say that a problem has **serializable subgoals** if there exists an order of subgoals such that the planner can achieve them in that order, without having to undo any of the previously achieved subgoals. For the Remote Agent planner which commanded NASA's Deep Space One spacecraft, it was determined that the propositions involved in commanding a spacecraft are serializable.

While perfectly decomposable problems are delicious but rare. The design of many planning systems—particularly the partial-order planners is based on the assumption that most real-world problems are **nearly decomposable**. Partial-order planning (POP) algorithms explore the space of plans without committing to a totally ordered sequence of actions. They work back from the goal, adding actions to the plan to achieve each subgoal. They are particularly effective on problems amenable to a divide-and-conquer approach.

The ideas underlying partial-order planning include the detection of conflicts and the protection of achieved conditions from interference. The construction of partially ordered plans (then called **task networks**) was pioneered by the NOAH planner [8] and by Tate's NONLIN system [9] . Partial-order planning dominated the next 20 years of research.

The heuristics to be used for total-order and partial-order planning can suffer from inaccuracies. A special data structure called a **planning graph** can be used to give better heuristic estimates. These heuristics can be applied to any of the search techniques including A* search described above. Alternatively, we can extract a solution directly from the planning graph, using a specialized algorithm such as the one called GRAPHPLAN [10].

Planners such as GRAPHPLAN, SATPLAN[11] and BLACKBOX [12] (GraphPlan + SatPlan) have moved the field of planning forward, both by raising the level of performance of planning systems and by clarifying the representational and combinatorial issues involved. These methods are, however, inherently propositional and thus are limited in the domains they can express (problems with no variables). For problems with large numbers of objects, this could result in a very substantial blowup in the number of action schemata. Despite this, planning graphs have proved to be effective tools for solving hard planning problems.

The jury is still out [7], but there are now some interesting comparisons of the various approaches to planning. Helmert (2001) analyzes several classes of planning problems, and shows that constraint-based approaches, such as GRAPHPLAN and SATPLAN are best for NP- hard domains, while search-based approaches do better in domains where feasible solutions can be found without backtracking. GRAPHPLAN and SATPLAN have trouble in domains with many objects, because that means they must create many actions. In some cases the problem can be delayed or avoided by generating the propositionalized actions dynamically, only as needed, rather than instantiating them all before the search begins.

References

[1] T. Bylander, The computational complexity of propositional STRIPS planning, Artificial Intelligence 69 (1994) 161–204.

[2] O. Madani, S. Hanks, A. Condon, On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems, in: Proceedings of the Sixteenth National Conference on Artificial Intelligence, AAAI Press/MIT Press, 1999, pp. 541–548.

[3] Contingent planning under uncertainty via stochastic satisfiability
http://ac.els-cdn.com/S000437020200379X/1-s2.0-S000437020200379X-main.pdf

[4] http://classes.engr.oregonstate.edu/eecs/spring2016/cs533/

[5] https://en.wikipedia.org/wiki/Timeline_of_artificial_intelligence

[6] https://en.wikipedia.org/wiki/Shakey_the_Robot

[7] Russell. S, and P. Norvig. Artificial Intelligence: A Modern Approach, Chapter 10: Classical Planning

[8] https://en.wikipedia.org/wiki/Partial-order_planning

[9] Tate, A. (1976) "Project Planning Using a Hierarchic Non-linear Planner", D.A.I. Research Report No. 25, August 1976, Department of Artificial Intelligence, University of Edinburgh.

[10] A. Blum and M. Furst (1997). Fast planning through planning graph analysis. Artificial intelligence. 90:281-300.

[11] http://www.cs.cornell.edu/selman/papers/pdf/92.ecai.satplan.pdf

[12] Henry Kautz and Bart Selman (1998). BLACKBOX: A New Approach to the Application of Theorem Proving to Problem Solving.